Package 'AR1seg'

February 19, 2015

Type Package

Title Segmentation of an autoregressive Gaussian process of order 1

Version 1.0

Depends Segmentor3IsBack

Date 2014-06-04

Author S. Chakar, E. Lebarbier, C. Levy-Leduc, S. Robin

Maintainer Souhil Chakar <souhil.chakar@agroparistech.fr>

Description This package corresponds to the implementation of the robust approach for estimating change-points in the mean of an AR(1) Gaussian process by using the methodology described in the paper arXiv 1403.1958

License GPL-2

NeedsCompilation no

Repository CRAN

Date/Publication 2014-06-05 15:58:15

R topics documented:

`AR1seg-package` *Segmentation of an AR(1) Gaussian process*

Description

This package consists in an implementation of a robust approach to solve the problem of multiple change-point estimation in the mean of a Gaussian AR(1) process. A robust estimator of the autoregression parameter is proposed and used to build a decorrelated series on which a classical penalized least-square approach is applied.

Details

Package:	AR1seg
Type:	Package
Version:	1.0
Date:	2014-06-04
License:	GPL-2

Author(s)

S. Chakar, E. Lebarbier, C. Levy-Leduc, S. Robin

Maintainer: Souhil Chakar <souhil.chakar@agroparistech.fr>

References

S. Chakar, E. Lebarbier, C. Levy-Leduc, S. Robin. A robust approach to multiple change-point estimation in an AR(1) process, arXiv:1403.1958.

Examples

```
library(AR1seg)
data(y)
res=AR1seg_func(y,Kmax=15,rho=TRUE)
a=c(1,res$PPSelectedBreaks[1:(res$PPselected-1)]+1)
b=res$PPSelectedBreaks[1:(res$PPselected)]
Bounds=cbind(a,b)
mu.fit=rep(res$PPmean,Bounds[,2]-Bounds[,1]+1)
plot(y)
lines(mu.fit,col="red")
```

`AR1seg_func` *Segmentation of an AR(1) Gaussian process*

Description

This function consists in an implementation of a robust approach to solve the problem of multiple change-point estimation in the mean of a Gaussian AR(1) process. A robust estimator of the autoregression parameter is proposed and used to build a decorrelated series on which a classical penalized least-square approach is applied.

Usage

```
AR1seg_func(y, Kmax = 15, rho = TRUE)
```

Arguments

y	Vector of observations
Kmax	Maximal number of segments
rho	It corresponds to the autoregression parameter. If it is equal to TRUE then it is estimated using a robust approach, otherwise the user has to provide a numerical value. By default, the value of rho is TRUE.

Value

Contains the following attributes:

data	Vector of observations
rho	The estimator of rho if the argument rho=TRUE, otherwise the value provided by the user
decorrelated	The decorrelated series using rho
breaks	Matrix of size Kmax*Kmax. The line K=1,...,Kmax corresponds to the optimal segmentation of the series with K segments. By convention, the last break of each line is the length of the series.
selected	Selected number of segments using the modified BIC criterion proposed by Zhang and Siegmund (2007)
SelectedBreaks	Optimal segmentation with a number of segments equal to the value selected
PPbreaks	Matrix of breaks obtained after the post-processing step
PPSelectedBreaks	Result of the post-processing step applied to SelectedBreaks: it is the resulting segmentation of our approach
PPselected	Length of the resulting segmentation (PPSelectedBreaks)
PPmean	Empirical mean of the series on each segment of the resulting segmentation

Note

This package depends on the package Segmentor3IsBack

Author(s)

S. Chakar, E. Lebarbier, C. Levy-Leduc, S. Robin

References

This function corresponds to the implementation of the robust approach for estimating change-points in the mean of an AR(1) Gaussian process by using the methodology described in the paper arXiv:1403.1958

Examples

```
##---- Should be DIRECTLY executable !! ----
##-- ==>  Define data, use random,
##--or do  help(data=index)  for the standard data sets.

## The function is currently defined as
function (y, Kmax = 15, rho = TRUE)
{
    l = length(y)
    if (rho)
        rho = median((diff(y, lag = 2))^2)/median(diff(y)^2) -
            1
    x = y[2:l] - rho * y[1:(l - 1)]
    S = Segmentor(x, model = 2, Kmax = Kmax)
    breaks = S@breaks
    for (i in 1:Kmax) {
        for (j in 1:i) breaks[i, j] = breaks[i, j] + 1
    }
    rm(i, j)
    parameters = S@parameters
    PP = function(t) {
        x = t
        l = length(x)
        i = 2
        while (l > 2 && i < l) {
            if (x[i] == x[i - 1] + 1 && x[i] != x[i + 1] - 1) {
                x = c(x[1:(i - 1)], x[(i + 1):l])
                l = l - 1
            }
            else i = i + 1
        }
        if (l > 1 && x[l - 1] == x[l] - 1)
            x = x[1:(l - 1)]
        x
    }
    PPbreaks = matrix(0, nrow = Kmax, ncol = Kmax, dimnames = dimnames(breaks))
    PPbreaks[1, ] = breaks[1, ]
    for (i in 2:Kmax) {
        t = PP(breaks[i, 1:(i - 1)])
        PPbreaks[i, ] = c(t, l, rep(0, Kmax - length(t) - 1))
    }
    rm(i, t)
    fMa = function(t, mu) {
        M = c()
        t = c(0, t)
        for (i in 2:length(t)) {
            M = c(M, rep(mu[i - 1], t[i] - t[i - 1]))
        }
        M
    }
    sswg = function(br, param, series) {
        sum((series - fMa(br, param))^2)
```

```
    }
    sswgseg = function(seg, seri) {
        res = c()
        for (i in 1:(Kmax)) {
            res = c(res, sswg(seg@breaks[i, 1:i], seg@parameters[i,
                1:i], seri))
        }
        res
    }
    minushalflogB = function(t, u) {
        t = t[t != 0]
        l = length(t)
        b = log(t[1]/u)
        if (l > 1) {
            for (i in 2:l) {
                b = b + log(t[i] - t[i - 1])
            }
        }
        b = -b/2
    }
    ZS = function(seg, seri) {
        u = length(seg@data)
        Kmax = seg@Kmax
        f = function(t) minushalflogB(t, u)
        wg = sswgseg(seg, seri)
        criterion = -(((u + 1):(u - Kmax + 2))/2) * log(wg) +
            lgamma(((u + 1):(u - Kmax + 2))/2) - (0:(Kmax - 1)) *
            log(u) + apply(seg@breaks, 1, f)
        selected = which.max(criterion)
        selected
    }
    selected = ZS(S, x)
    SelectedBreaks = breaks[selected, 1:selected]
    PPSelectedBreaks = PPbreaks[selected, ]
    PPSelectedBreaks = PPSelectedBreaks[PPSelectedBreaks != 0]
    PPselected = length(PPSelectedBreaks)
    vec1 = c(1, PPSelectedBreaks[1:(PPselected - 1)] + 1)
    vec2 = PPSelectedBreaks[1:(PPselected)]
    m = c()
    for (i in 1:PPselected) {
        m[i] = mean(y[vec1[i]:vec2[i]])
    }
    list(data = y, rho = rho, decorrelated = x, breaks = breaks,
        PPbreaks = PPbreaks, selected = selected, SelectedBreaks = SelectedBreaks,
        PPSelectedBreaks = PPSelectedBreaks, PPselected = PPselected, PPmean = m)
}
```

---

y *Vector of observations*

Description

Vector of size 1600 which is a piecewise constant function corrupted by a Gaussian AR(1) process with rho=0.3 and where the standard deviation of the innovation process is equal to 0.1. The piecewise constant function has its breaks at the following positions: 222, 311, 711, 888, 1200, 1466 and takes the following values: 0, 1, 0, 1, 0, 1, 0 on each associated segment.

Usage

```
data(y)
```

Format

The format is: num [1:1600] 0.11834 0.02428 0.00802 0.06716 0.10555 ...

Examples

```
library(AR1seg)
data(y)
```

Index